

# Transform Flow: A Mobile Augmented Reality Visualisation and Evaluation Toolkit

Samuel Williams  
Computer Science  
University of Canterbury  
Christchurch, New Zealand

Richard Green  
Computer Science  
University of Canterbury  
Christchurch, New Zealand

Mark Billinghurst  
The HIT Lab NZ  
University of Canterbury  
Christchurch, New Zealand

**Abstract**—It is common in mobile augmented reality (AR) research to examine results that were attained with unpublished tools and data sets, which makes it difficult to compare and improve existing work without significant effort. We discuss the development of an open source toolkit called Transform Flow, which includes a data capture application for iOS, a desktop application to replay and analyse captured data sets with different algorithms, and a mobile application that can run these algorithms in real-time. Our results suggest that our toolkit can be a centre for collaborative research, as it provides a common platform on which tracking algorithms for mobile AR can be developed, studied and eventually deployed.

## I. INTRODUCTION

Evaluating and comparing algorithms is an important part of quality research. The ability to compare different approaches with the same data sets is critical to the development of improved methods. This is also true for mobile augmented reality (AR), where many tracking algorithms are developed specifically within the constraints of a particular hardware configuration and testing methodology.

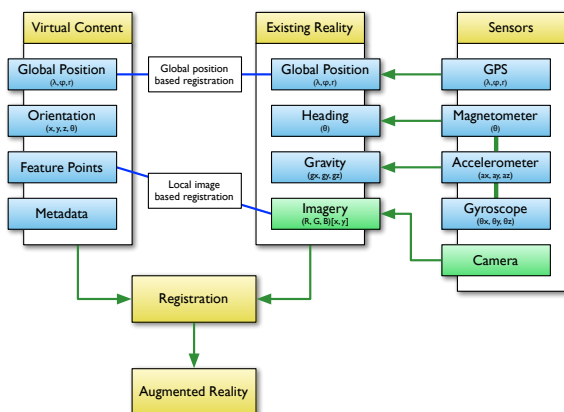


Fig. 1. An overview of typical data flow in mobile outdoor AR.

Ideally, sensor and visual data (see figure 1) is captured from a variety of mobile devices in such a way that it can be repeatedly replayed. Algorithms can be critically analysed, frame by frame, to isolate edge cases and bugs in a controlled environment. Test cases can be developed and run automatically to check for regressions. Once perfected,

the code can be easily adapted for implementation on a mobile device.

An open source platform that facilitates the analysis of new and existing algorithms would allow researchers to develop and test ideas easily. Existing code can be reused, which typically reduces the challenges and risks associated with software development. Such a platform could also serve as a useful educational tool for students wanting to learn about different tracking algorithms and how they are implemented.

We describe the ongoing development of Transform Flow<sup>1</sup>, which has recently been released under the MIT license. It was originally developed as part of a single research project, but is being released to the community to encourage collaborative research, development and education in the area of mobile AR.

## II. BACKGROUND

Testing and evaluating tracking algorithms designed for mobile devices is currently unsystematic[1], [2], [3], [4]. Algorithms are often structured around specific platforms and hardware. Similarly, existing public data sets that don't include inertial sensor measurements[5], [6] are not appropriate for modern mobile AR research[7].

Even as recently as 2007, researchers were developing custom hardware for outdoor AR[8]. The difficulty of reproducing such devices makes it practically impossible to recreate published results. In contrast, modern consumer mobile phones and tablets provide an excellent variety of sensors in low cost, readily available platforms. It is typical to have access to gyroscopes, accelerometers, magnetometers, global positioning and cameras. However, despite this level of standardisation, we lack modern tools and data sets for testing and evaluating tracking algorithms. Practical implementations need to deal with a wide variety of sensor accuracies and hardware tolerances.

Published research in mobile AR often lacks source code[4], [7]. There are many reasons for this, but practically speaking, it makes the development and testing of new approaches difficult. Data sets and testing tools are often not publicly published[9] which makes it difficult to check whether a new approach is a significant improvement over existing methods. In particular, algorithms that fail

<sup>1</sup> Available online: <https://github.com/HITLabNZ/transform-flow>

on specific edge cases[4] warrant further analysis and study; but without specific data sets and implementations this is impossible.

There are several open source projects which implement inertial sensor based mobile AR tracking algorithms. However, many of the most promising ones seem unmaintained[10], [11], [12]. Other libraries which are maintained, are device specific[13] and application specific[14]. None of the evaluated libraries provide specific tools for the research and development of mobile outdoor AR tracking algorithms.

### A. Motivations

The software described in this paper was developed to support a research project investigating improved methods for outdoor AR tracking. It represents a significant investment of time. We have decided to publish the code, documentation, and data sets as completely unencumbered open source, in the hopes that its availability will reduce the barriers we faced when first entering this field.

In addition, we believe that the structure of our contribution can encourage a systematic approach to the development of new algorithms. The existing code base includes many practical examples, and over time we hope that our tooling can serve as a useful platform for a wide variety of tracking implementations.

This is an ongoing project and we realise that it may not fit everyone’s requirements. We have specifically chosen online systems that support collaboration, so as to maximise the value of the toolkit - not just for our needs - but for the needs of this field in general.

## III. DATA CAPTURE

We created a mobile data acquisition system[15] to capture sensor data and image frames. This application was developed for iOS using Objective-C. The rate at which sensor data and video frames are captured is independent, and can be customised at compile time. All data is saved in a CSV (comma separated values) formatted log file, including video frames that reference external PNG (portable network graphics) image files in the same directory.

Sensor data is captured using CoreLocation to access the location (via GPS) and heading (via compass), and CoreMotion to access the accelerometer, gyroscope and gravity vectors. The gravity vector is already provided by most modern platforms including both Android and iOS, and there is the chance that it is optimised in hardware, so we use it directly.

Video frames are captured using AVFoundation that provides access to the camera at a variety of resolutions. We record the captured image as interpolated RGB, which is universally supported across all devices and a convenient format for further processing.

Phone specific data including the device name and hardware identification are recorded as one of the first



Fig. 2. The data acquisition application running on an iPhone 5.

entries in the log file. This can be used for hardware or device specific calibration files.

The tool includes a real-time visualisation of sensor measurements. This allows for a greater understanding of the effects that the physical device motion is having on the sensor data. It can be instructive to check the gyroscope, accelerometer or other sensors while manipulating the device: per-device issues including calibration problems and drift can be assessed quickly and isolated.

### A. Data Set Format

Data sets are recorded using the on-screen switch. This information is saved into the phone’s memory and can be downloaded to a computer using Xcode. The data sets themselves were designed to be simple to work with and flexible enough to support future requirements.

We use CSV as it is a simple format for structured data and can be logged easily. We uniquely identify each row in the log using an index starting from 1, a function name (e.g. “Gyroscope”) and a timestamp. The function name relates to the structure of the remaining arguments in the row and is used for processing rows into useful data structures.

It is easy to modify the data capture tool to add additional data structure records, e.g. adding battery status, GPS course, current waypoint, or other parameters that may be of interest. This allows for the data capture application to be extended with new capabilities as required for specific research areas, while retaining core functionality and compatibility with existing processing/visualisation tools.

Nested data can be supported using sequential relationships. For ease of readability, gyroscope, accelerometer and gravity vectors are recorded separately, but are immediately followed by a motion entry with the same timestamp, which ties them together into one logical unit for processing. In the future, additional motion specific fields could be added, e.g. magnetometer measurements, without modifying any other structures.

## IV. VISUALISATION

We have developed a desktop application[16] for viewing data sets captured using the mobile data acquisition

tool. This application is a general-purpose C++11 application using OpenGL for rendering. It currently compiles for Mac OS X and Linux support is on the way.

Our current implementation assumes that the user is interested in visualising data within a global frame of reference, with position defined by (*latitude, longitude, altitude*) and rotation defined by (*bearing, gravity*). To systematically capture this data, we define an abstract motion model that is used to process incoming sensor data and image frames.

The input data set, combined with a motion model, produces an immutable per-frame globally registered camera pose. This sequence of frames can then be visualised and analysed without further motion processing.

Each frame can have a set of debug notes and visual markers associated with it. Per-frame logging is a useful debugging tool and provides specific feedback about how the motion model was working. Per-frame feature points show a structured analysis of the image data and allow a user analyse the details of individual features by selecting them repeatedly.

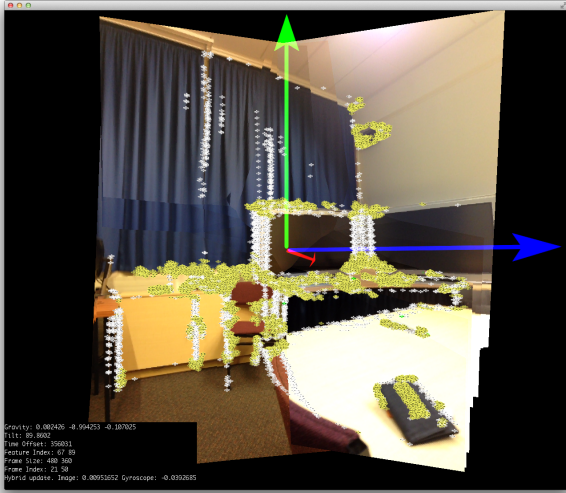


Fig. 3. The data visualisation application rendering 50 combined frames.

Displaying frames in a 3D environment is challenging because a camera frame is a planar projection of a typically non-planar environment. Without depth information it is impossible to accurately reproduce the actual 3D structure. To work around this problem, the visualisation tool uses planar projections based on the camera’s field of view and an arbitrary scale factor. Frames can then be rendered in a 3D environment and explored using an arbitrary view position. Pure rotations result in the easiest to interpret visualisation as there is no change in planar alignment.

#### A. Motion Models

A motion model is an algorithm or set of algorithms that implement a tracking algorithm. The set of inputs and outputs are well defined, and relate specifically to the tracking task being performed. We specifically designed

our motion model abstraction around the hardware capabilities of modern mobile devices and the types of data required for accurate global registration.

```
class MotionModel
{
public:
    // Inputs:
    virtual void update(const LocationUpdate &) = 0;
    virtual void update(const HeadingUpdate &) = 0;
    virtual void update(const MotionUpdate &) = 0;
    virtual void update(const ImageUpdate &) = 0;

    // Outputs:
    virtual bool localization_valid() const;
    const Vec3 & gravity() const;
    const Vec3 & position() const;
    Radians<> bearing() const;
};
```

Specific motion models may require additional inputs or outputs. For local image based tracking, we may output a camera pose relative a local frame of reference. Other motion models might require direct access to the magnetometer. This is supported by subclassing and modifying the source code appropriately.

To serve as an example, we include two motion models in the current distribution. These motion models can be compiled and used both in the visualisation tool and in the browser application:

1) *Basic Sensor Motion Model*: The included basic sensor motion model is a very simple implementation of sensor fusion. It combines the compass and gyroscope using a low-pass filter for improved bearing calculations.

In order to initialise, it requires at least one motion update to establish a gravity direction, and one heading update to establish a global bearing. After that point, gyroscope updates are combined with heading. Along with the gravity vector, this provides a relatively robust rotational frame of reference.

2) *Hybrid Motion Model*: The hybrid motion model derives from the basic sensor motion model and incorporates image processing into the bearing calculation. After it has received at least one image update, each subsequent image update can be used to correct changes in bearing.

#### B. Analysis

The visualisation tool provides a number of methods for evaluation. Rotational distribution seeks to provide a systematic way to measure the accuracy of pure rotation based motion. Translational distribution aims to provide a systematic way to measure the accuracy in a general six degree of freedom (6DOF) data set.

Methods for analysis are currently ad-hoc - meaning that the user must manually select points. However, we hope that in the future standardised evaluation methods can be codified into the framework and work directly with the output from the motion model.



### C. Sample Data Sets

In order to facilitate systematic testing and evaluation, and additionally provide sample data for the visualisation tool, several data sets have been published and documented[17]. These data sets are captured using the published iOS capture tool, and include a full range of sensor measurements. We hope the publication of these data sets will stimulate others to do the same and that over time we can build up a large corpus of sample data to support a wide range of systematic evaluation techniques.

## V. DEPLOYMENT

We have developed an iOS application[18] that can be used to run algorithms developed using the Transform Flow motion model abstraction. It uses a similar setup to the capture tool, but rather than logging the events, it applies them directly to a motion model. The AR visualisation is then rendered using the calculated frame of reference.

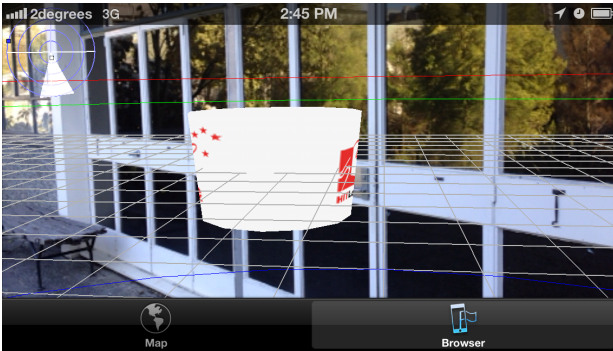


Fig. 4. ARBrowser running using the Transform Flow Basic Sensor Motion Model.

The application itself consists of two parts, a map view and an AR browser (see figure 4). The map view shows the user location and nearby globally registered points of interest. The browser can display points of interest using 3D content (via Wavefront OBJ files) and 2D billboards (constructed directly from UIView instances), which are rendered using OpenGL ES 1.0.

Dealing with a wide variety of camera and screen configurations is not trivial. Specifically, different cameras have different intrinsic properties, the most important being the field of view. We estimate this parameter as  $55^\circ$  which is in practice  $\pm 2^\circ$  for commonly used iOS devices. Device specific calibrations might be useful in a research setting but would be cumbersome for end user applications. Ideally, a per-device database of camera intrinsic properties would allow for wide spread deployment with acceptable accuracy.

## VI. SOURCE CODE

One of the important goals of this project was to create something that would facilitate collaboration and further research. As such, the data capture, analysis and browser tools have been released under the MIT license on GitHub. The MIT license allows developers to use the source code

free of charge with very few limitations (e.g. commercial use is acceptable).

GitHub provides a fantastic environment based on Git[19] where other researchers can easily fork the source code and contribute back their modifications. We hope to integrate new motion models and evaluation methodologies as they are developed, so that our project can serve as a useful tool for comparative analysis for future work.

### A. Building

Existing research[20] and practical experience has shown that build related challenges can be a major source of problems:

- Choosing appropriate compilers and SDKs for the target platform.
- Selecting compiler flags in a way that is consistent across all systems.
- Specifying the directories where headers and libraries from other required software packages are located.
- Generating code and copying resources, e.g. shaders and sample data.
- Specifying the location(s) to produce object code, libraries, executables and install packages.

Originally we were using CMake[21]. As the project got larger and included more dependencies, the amount of CMake configuration required became unmanageable. Cross platform builds were difficult, especially when using modern C++ compilers, and these issues hindered the development of modular extensible software.

In order to simplify the build process, we developed a Ruby based build tool called Teapot[22] that allows us to fetch and compile versioned dependencies for the target platform. This includes libraries such as OpenCV (for computer vision algorithms) and Dream (for modern cross platform OpenGL). The inspiration for this tool came from RubyGems[23], which is essentially a platform-independent package management system. By automatically building dependencies, we reduce the burden of installing and updating the software.

### B. Unit Testing

An important part of a collaborative project is ensuring that changes and modifications don't introduce problems with existing code. GitHub, when combined with Travis-CI[24], provides immediate feedback when users make source code submissions (pull requests) on public projects.

Many component parts of the Transform Flow toolkit are unit tested, which ensures that new users making contributions can feel confident that they are not breaking existing code. Unit tests also provide useful examples of how to interact with various parts of the system. We hope this will allow new users to become familiar with the code quickly and easily.

### C. Android Support

Several researchers are presently working on Android support for both the data capture and browser applications. As the applications are platform specific, they require custom code for interacting with hardware (e.g. cameras, sensors) and the user (e.g. user interface). This means that there is a moderate amount of per-platform work required.

The base Transform Flow library[25] is written in modern C++11 with few platform specific dependencies. The supporting libraries are capable of being cross-compiled, which should ensure a seamless deployment for Transform Flow based algorithms across a wide range of devices with minimal (ideally zero) per-platform code required.

## VII. CONCLUSION

The data capture and visualisation tools are useful for developing, analysing and evaluating mobile outdoor AR tracking algorithms. We believe that they can reduce the time it takes to investigate new ideas and will ultimately improve the collaboration between existing researchers.

The AR browser client application can be used to validate tracking algorithms on a variety of real world devices and provides a starting point for usability testing and application development.

We have confidence in the long term viability of the project and will continue to maintain it for the foreseeable future. To support this, we implemented an open source development methodology which encourages collaboration and shared responsibility. We look forward to seeing how it develops over the next few years, and hope to see it used in exciting new research.

### A. Future Directions

We would like to develop several completely systematic methods of evaluation, improve the structure of the visualisations, and support more platforms.

## REFERENCES

- [1] S. Lee, Y.-H. Seo, and H. S. Yang, "Scalable building facade recognition and tracking for outdoor augmented reality," in *Information Technology Convergence*. Springer, 2013, pp. 923–931.
- [2] G. Klein and D. Murray, "Parallel tracking and mapping on a camera phone," in *Proceedings of the 2009 8th IEEE International Symposium on Mixed and Augmented Reality*, ser. ISMAR '09. Washington, DC, USA: IEEE Computer Society, 2009, pp. 83–86. [Online]. Available: <http://dx.doi.org/10.1109/ISMAR.2009.5336495>
- [3] G. Reitmayr and T. W. Drummond, "Going out: Robust tracking for outdoor augmented reality," in *Proc. ISMAR 2006*, IEEE and ACM. Santa Barbara, CA, USA: IEEE CS, October 22–25 2006, pp. 109–118.
- [4] G. Reitmayr, T. Langlotz, D. Wagner, A. Mulloni, G. Schall, D. Schmalstieg, and Q. Pan, "Simultaneous localization and mapping for augmented reality," in *Proceedings of the 2010 International Symposium on Ubiquitous Virtual Reality*, ser. ISUVR '10. Washington, DC, USA: IEEE Computer Society, 2010, pp. 5–8. [Online]. Available: <http://dx.doi.org/10.1109/ISUVR.2010.12>
- [5] S. Gauglitz, T. Höllerer, and M. Turk, "Evaluation of interest point detectors and feature descriptors for visual tracking," *Int. J. Comput. Vision*, vol. 94, no. 3, pp. 335–360, Sep. 2011. [Online]. Available: <http://dx.doi.org/10.1007/s11263-011-0431-5>
- [6] S. Lieberknecht, S. Benhimane, P. Meier, and N. Navab, "A dataset and evaluation methodology for template-based tracking algorithms," in *Proceedings of the 2009 8th IEEE International Symposium on Mixed and Augmented Reality*, ser. ISMAR '09. Washington, DC, USA: IEEE Computer Society, 2009, pp. 145–151. [Online]. Available: <http://dx.doi.org/10.1109/ISMAR.2009.5336487>
- [7] D. Kurz and S. Benhimane, "Gravity-aware handheld augmented reality," in *Proceedings of the 2011 10th IEEE International Symposium on Mixed and Augmented Reality*, ser. ISMAR '11. Washington, DC, USA: IEEE Computer Society, 2011, pp. 111–120. [Online]. Available: <http://dx.doi.org/10.1109/ISMAR.2011.6092376>
- [8] G. Reitmayr and T. Drummond, "Initialisation for visual tracking in urban environments," in *Proc. ISMAR 2007*, Nara, Japan, Nov. 13–16 2007, pp. 161–160.
- [9] D. Wagner, A. Mulloni, T. Langlotz, and D. Schmalstieg, "Real-time panoramic mapping and tracking on mobile phones," in *Proceedings of the 2010 IEEE Virtual Reality Conference*, ser. VR '10. Washington, DC, USA: IEEE Computer Society, 2010, pp. 211–218. [Online]. Available: <http://dx.doi.org/10.1109/VR.2010.5444786>
- [10] D. Schmalstieg, A. Fuhrmann, G. Hesina, Z. Szalavári, L. M. Encarnação, M. Gervautz, and W. Purgathofer, "The studierstube augmented reality project," *Presence: Teleoperators and Virtual Environments*, vol. 11, no. 1, pp. 33–54, 2002.
- [11] G. Reitmayr and D. Schmalstieg, "Opentracker: A flexible software design for three-dimensional interaction," *Virtual reality*, vol. 9, no. 1, pp. 79–92, 2005.
- [12] "LibreGeoSocial." [Online]. Available: <http://www.libregeosocial.org/>
- [13] "DroidAR." [Online]. Available: <https://github.com/bitstars/droidar>
- [14] "mixare." [Online]. Available: <http://www.mixare.org>
- [15] "Tranform Flow capture application for iOS." [Online]. Available: <https://github.com/HITLabNZ/transform-flow-capture-ios>
- [16] "Tranform Flow visualisation application." [Online]. Available: <https://github.com/HITLabNZ/transform-flow-visualisation>
- [17] "Tranform Flow data sets." [Online]. Available: <https://github.com/HITLabNZ/transform-flow-data>
- [18] "Tranform Flow browser application for iOS." [Online]. Available: <https://github.com/HITLabNZ/transform-flow-browser-ios>
- [19] L. Torvalds and J. Hamano, "Git: Fast version control system," 2010. [Online]. Available: <http://git-scm.com>
- [20] W. J. Schroeder, L. Ibáñez, and K. M. Martin, "Software process: the key to developing robust, reusable and maintainable open-source software," in *Biomedical Imaging: Nano to Macro, 2004. IEEE International Symposium on*. IEEE, 2004, pp. 648–651.
- [21] "CMake - cross platform make." [Online]. Available: <http://www.cmake.org>
- [22] "Teapot." [Online]. Available: <http://www.kyusu.org>
- [23] "RubyGems." [Online]. Available: <http://rubygems.org>
- [24] "Travis CI - free hosted continuous integration platform." [Online]. Available: <https://travis-ci.org>
- [25] "Tranform Flow library." [Online]. Available: <https://github.com/HITLabNZ/transform-flow>